

Creating and Modifying a Boxfill Graphics Method

Â

[Contents](#) [Previous](#) [Next](#)

Goal: Guide you through creating and setting boxfill graphics method attributes.

Before running the tutorial below, type "*python*" or "*cdat*" at the command line.Â You will see the python prompt appear (i.e., ">>>"). You can now enter the command lines below.

You can [view](#)Â or [download](#)Â the full source code. To run the source code at the command line, type: "*python boxfill_file.py*".

```
# Import the modules needed for the tutorial
# cdms - Climate Data Management system accesses gridded data.
# vcs - Visualization and control System 1D and 2D plotting routines.
# cdutil - Climate utilitizes that contains miscellaneous routines for
#           manipulating variables.
# time - This module provides various functions to manipulate time values.
# os - Operation System routines for Mac, DOS, NT, or Posix depending on
#       the system you're on.
# sys - This module provides access to some objects used or maintained by
#       the interpreter and to functions that interact strongly with the interpreter.
import vcs, cdms, cdutil, time, os, sys

# Open data file:
filepath = os.path.join(sys.prefix, 'sample_data/clt.nc')
cdmsfile = cdms.open( filepath )

# Extract a 3 dimensional data set and get a subset of the time dimension
data = cdmsfile('clt', longitude=(-180, 180), latitude = (-90., 90.))

# Initial VCS:
v = vcs.init()
```

The VCS module contains a list of persistent boxfill graphics method objects. To view this list issue the "show" command.

```
# Show the list of persistent boxfill graphics methods.
v.show('boxfill')
```

```
*****Boxfill Names List*****
( 1):          ASD          ASD_map          b_and_w
( 4):          default      polar            quick
( 7):          robinson
*****End Boxfill Names List*****
```

Get a boxfill graphics method object and plot:

```
# Assign the variable "bf_asd" to the persistent 'ASD' boxfill graphics methods.
bf_asd = v.getboxfill( 'ASD' )

# Plot the data using the above boxfill graphics method.
v.plot( data, bf_asd )
```

List the 'ASD' boxfill graphics method attributes by issuing the following command:

```
# List the 'ASD' boxfill graphics methods attributes.  
bf_asd.list()  
  
-----Boxfill (Gfb) member (attribute) listings -----  
Canvas Mode = 1  
graphics method = Gfb  
name = ASD  
projection = linear  
xticlabels1 = *  
xticlabels2 = *  
xmtics1 =  
xmtics2 =  
yticlabels1 = *  
yticlabels2 = *  
ymtics1 =  
ymtics2 =  
datawc_x1 = 1.00000002004e+20  
datawc_y1 = 1.00000002004e+20  
datawc_x2 = 1.00000002004e+20  
datawc_y2 = 1.00000002004e+20  
datawc_timeunits = days since 2000  
datawc_calendar = 135441  
xaxisconvert = linear  
yaxisconvert = linear  
boxfill_type = linear  
level_1 = 1.00000002004e+20  
level_2 = 1.00000002004e+20  
levels = ([1.0000000200408773e+20, 1.0000000200408773e+20],)  
color_1 = 16  
color_2 = 239  
fillareacolors = None  
legend = None  
ext_1 = n  
ext_2 = n  
missing = 241
```

```
# Change 'ASD' boxfill graphics methods attributes by entering the following commands.  
bf_asd.level_1 = 20          # set the minimum data value  
bf_asd.level_2 = 80          # set the maximum data value  
bf_asd.legend = {25:'Blue', 40:'Green', 58:'Yellow', 77:'Red'}  
bf_asd.color_1 = 40           # change the 1st color index  
bf_asd.color_2 = 228          # change the 2nd color index  
bf_asd.datawc(-45.0, 45.0, -90.0, 90.0)  # change the region
```

Generate a \log_{10} boxfill plot:

```
# Generate a log10 boxfill graphics method plot  
bf_asd.boxfill_type='log10' # change the boxfill type to log10  
bf_asd.datawc(1e20,1e20,1e20,1e20) # change to default region  
bf_asd.level_1=1e20          # change to default minimum level  
bf_asd.level_2=1e20          # change to default maximum level  
bf_asd.color_1=16            # change 1st color index value  
bf_asd.color_2=239           # change 2nd color index value
```

Generate a custom boxfill plot

```
# Generate a custom boxfill graphics method plot  
bf_asd.boxfill_type='custom' # change the boxfill type to custom  
bf_asd.levels=(0,20,35,40,75,100) # set the custom ranges  
bf_asd.fillareacolors=(16,55,155,200,235) # set the color indices
```

Create your own boxfill graphics method:

```
# Create a persistent boxfill graphics methods from an existing boxfill graphics method.  
bf_new = v.createboxfill( 'new', 'ASD' ) # create new from ASD  
bf_new2 = v.createboxfill( 'new2','quick')# create new2 from quick  
bf_new.color_1=50 # change color level  
bf_new.list() # list its attributes  
v.show('boxfill') # show boxfill list with new and new2
```

```
*****Boxfill Names List*****  
( 1): ASD ASD_map b_and_w  
( 4): default new new2  
( 7): polar quick robinson  
*****End Boxfill Names List*****
```

```
v.removeobject( bf_new ) # remove new from boxfill list  
v.show('boxfill') # show boxfill list without new
```

```
*****Boxfill Names List*****  
( 1): ASD ASD_map b_and_w  
( 4): default new2 polar  
( 7): quick robinson  
*****End Boxfill Names List*****
```

Â

[Contents](#) [Previous](#) [Next](#)